Two-Stage Transfer Surrogate Model for Automatic Hyperparameter Optimization

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab
University of Hildesheim
Universitätsplatz 1, 31141 Hildesheim, Germany
{wistuba,schilling,schmidt-thieme}@ismll.uni-hildesheim.de

Abstract. The choice of hyperparameters and the selection of algorithms is a crucial part in machine learning. Bayesian optimization methods have been used very successfully to tune hyperparameters automatically, in many cases even being able to outperform the human expert. Recently, these techniques have been massively improved by using meta-knowledge. The idea is to use knowledge of the performance of an algorithm on given other data sets to automatically accelerate the hyperparameter optimization for a new data set.
In this work we present a model that transfers this knowledge in two stages. At the first stage, the function that maps hyperparameter configurations to hold-out validation performances is approximated for previously seen data sets. At the second stage, these approximations are combined to rank the hyperparameter configurations for a new data set. In extensive experiments on the problem of hyperparameter optimization as well as the problem of combined algorithm selection and hyperparameter optimization, we are outperforming the state of the art methods.

Keywords: hyperparameter optimization, meta-learning, transfer learning

1 Introduction

The tuning of hyperparameters is an omnipresent problem in the machine learning community. In comparison to model parameters, which are estimated by a learning algorithm, hyperparameters are parameters that have to be specified before the execution of the algorithm. Typical examples for hyperparameters are the trade-off parameter C of a support vector machine or the number of layers and nodes in a neural network. Unfortunately, the choice of the hyperparameters is crucial and decides whether the performance of an algorithm is state of the art or just moderate. Hence, the task of hyperparameter optimization is as important as developing new models [2, 5, 18, 23, 27].

The traditional way of finding good hyperparameter configurations is by using a combination of manual and grid search. This procedure are a brute force approach of searching the hyperparameter space. They are very time-consuming

or even infeasible for high-dimensional hyperparameter spaces. Therefore, methods to steer the search for good hyperparameter configurations are currently an interesting topic for researchers [3, 23, 27].

Sequential model-based optimization (SMBO) [13] is a black-box optimization framework and is currently the state of the art for automatic hyperparameter optimization. Within this framework, the trials of already tested hyperparameter configurations are used to approximate the true hyperparameter response function using a surrogate model. Based on this approximation, a promising new hyperparameter configuration is chosen and tested in the next step. The result of this next trial is then used to update the surrogate model for further hyperparameter configuration acquisitions.

Human experts utilize their experience with a machine learning model and try hyperparameter configurations that have been good on *other* data sets. This transfer of knowledge is one important research direction in the domain of automatic hyperparameter optimization. Currently, two different approaches to integrate this idea into the SMBO framework exist. Either by training the surrogate model on past experiments [1, 21, 25, 33], or by using the information on past experiments to initialize the new search [7, 32].

We propose a two-stage approach to consider the experiences with different hyperparameter configurations on other data sets. At the first stage, we approximate the hyperparameter response function of the new data set as well as of previous data sets. This approximation is then combined to rank the hyperparameter configurations for the new data set, considering the similarity between the new data set and the previous ones. In two extensive experiments for the problem of hyperparameter optimization and the problem of combined algorithm selection and hyperparameter optimization, we show that our two-stage approach is able to outperform current state of the art competitor methods, which have been recently published on established machine learning conferences.

2 Related Work

The aim of automatic hyperparameter optimization is to enable non-experts to successfully use machine learning models but also to accelerate the process of finding good hyperparameter configurations. Sequential model-based optimization (SMBO) is the current state of the art for automatic hyperparameter optimization. Various approaches exist to accelerate the search for good hyperparameter configurations. One important approach is the use of meta-knowledge. This approach has already proven its benefit for other hyperparameter optimization approaches [9, 16, 20, 29]. One easy way to make use of meta-knowledge is through initialization. This approach is universal and can be applied for every hyperparameter optimization method. Reif et al. [20] suggest to choose those hyperparameter configurations for a new data set as initial trials that performed best on a similar data set in the context of evolutionary parameter optimization. Here, the similarity was defined through the distance among meta-features, which describe properties of a data set. This idea was applied to SMBO by Feurer

et al. [7] and later improved [8]. Recently, it was proposed to learn initial hyperparameter configurations in such a way that it is no longer necessary to be limited to choose initial hyperparameter configurations from the set of hyperparameter configurations, which have been chosen in previous experiments [32].

While the initialization can be used for any hyperparameter optimization method, the idea to use transfer surrogate models is specific for the SMBO framework. Bardenet et al. [1] were the first who proposed to learn the surrogate model not only on the current data set but also over previous experiments in order to make use of the meta-knowledge. Soon, this idea was further investigated: specific Gaussian processes [25, 33] and neural networks [21] were proposed as surrogate models.

The aforementioned ideas make use of meta-knowledge to accelerate the search for good hyperparameter configurations. Another way of saving time is to stop an hyperparameter configuration evaluation early if it appears to be not promising after few training iterations. Obviously, this is only possible for iterative learning algorithms, which are using gradient-based optimization. Even though this approach is orthogonal to the meta-learning approach, the aim is the same, i.e. accelerating the search for good hyperparameter configurations. Domhan et al. [6] propose to predict the development of the learning curve based on few iterations. If the predicted development is less promising than the currently best configuration, the currently investigated configuration is discarded. A similar approach is proposed by Swersky et al. [26]. Instead of trying different configurations sequentially and eventually discarding them, they learn the models for various hyperparameter configurations at the same time and switch from one learning process to the other if it looks more promising.

3 Background

In this section the hyperparameter optimization problem is formally defined and, for the sake of completeness, the sequential model-based optimization framework is presented.

3.1 Hyperparameter Optimization Problem Setup

A machine learning algorithm $\mathcal{A}_{\boldsymbol{\lambda}}$ is a mapping $\mathcal{A}_{\boldsymbol{\lambda}} : \mathcal{D} \rightarrow \mathcal{M}$ where $\mathcal{D}$ is the set of all data sets, $\mathcal{M}$ is the space of all models and $\boldsymbol{\lambda} \in \Lambda$ is the chosen hyperparameter configuration with $\Lambda = \Lambda_1 \times \ldots \times \Lambda_P$ being the P-dimensional hyperparameter space. The learning algorithm estimates a model $M_{\boldsymbol{\lambda}} \in \mathcal{M}$, which minimizes a loss function $\mathcal{L}$ (e.g. residual sum of squares), that is penalized with a regularization term $\mathcal{R}$ (e.g. Tikhonov regularization) with respect to the training set D^{train} of the data set D:

$$\mathcal{A}_{\boldsymbol{\lambda}}\left(D^{\text{train}}\right) = \arg\min_{M_{\boldsymbol{\lambda}} \in \mathcal{M}} \mathcal{L}\left(M_{\boldsymbol{\lambda}}, D^{\text{train}}\right) + \mathcal{R}\left(M_{\boldsymbol{\lambda}}\right). \tag{1}$$

Then, the task of *hyperparameter optimization* is to find the hyperparameter configuration $\boldsymbol{\lambda}^*$ that leads to a model $M_{\boldsymbol{\lambda}^*}$, which minimizes the loss on the

validation data set D^{valid}, i.e.

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \Lambda} \mathcal{L}\left(\mathcal{A}_{\boldsymbol{\lambda}}\left(D^{\text{train}}\right), D^{\text{valid}}\right) = \arg\min_{\boldsymbol{\lambda} \in \Lambda} f_D\left(\boldsymbol{\lambda}\right). \tag{2}$$

The function f_D is the *hyperparameter response function* of data set D.

$$f_D\left(\boldsymbol{\lambda}\right) = \mathcal{L}\left(\mathcal{A}_{\boldsymbol{\lambda}}\left(D^{\text{train}}\right), D^{\text{valid}}\right) \tag{3}$$

For the sake of demonstration, in the remaining sections, we consider the problem of tuning the hyperparameters of a classifier. Thus, f returns the misclassification rate. This is obviously no limitation, but shall help the reader to understand the concepts given a concrete example.

3.2 Sequential Model-based Optimization

Sequential model-based optimization (SMBO) [13], originally proposed for black-box optimization, can be used for optimizing hyperparameters automatically by using the SMBO framework to minimize the hyperparameter response function (Equation 3) [2]. SMBO consists of two components, i) a surrogate model Ψ, that is used to approximate the function f, which we want to minimize, and ii) an acquisition function a, that decides which hyperparameter to try next.

Algorithm 1 outlines the SMBO framework for minimizing the function f. For T many iterations different hyperparameters are tried. In iteration t, we approximate f using our surrogate model Ψ_{t+1} based on the observation history $\mathcal{H}_t$, the set of all hyperparameter configurations and performances, which have been evaluated evaluated so far. The surrogate model is an approximation of f with the property that it can be evaluated fast. Based on the predictions of Ψ and the corresponding uncertainties about these predictions, the acquisition function finds a trade-off between exploitation and exploration and determines the hyperparameter configuration to try next. This configuration is then evaluated, and the new observation is added to the observation history. After T trials, the best performing hyperparameter configuration is returned.

Algorithm 1 Sequential Model-based Optimization

Input: Hyperparameter space Λ, observation history $\mathcal{H}$, number of trials T, acquisition function a, surrogate model Ψ.
Output: Best hyperparameter configuration found.

for $t = 1$ to T **do**
  Fit Ψ_{t+1} to $\mathcal{H}_t$
  $\boldsymbol{\lambda} \leftarrow \arg\max_{\boldsymbol{\lambda} \in \Lambda} a\left(\mu\left(\Psi_{t+1}\left(\boldsymbol{\lambda}\right)\right), \sigma\left(\Psi_{t+1}\left(\boldsymbol{\lambda}\right)\right), f^{\min}\right)$
  Evaluate $f\left(\boldsymbol{\lambda}\right)$
  $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \cup \{(\boldsymbol{\lambda}, f\left(\boldsymbol{\lambda}\right))\}$
  if $f\left(\boldsymbol{\lambda}\right) < f^{\min}$ **then**
    $\boldsymbol{\lambda}^{\min}, f^{\min} \leftarrow \boldsymbol{\lambda}, f\left(\boldsymbol{\lambda}\right)$
return $\boldsymbol{\lambda}^{\min}$

Since the acquisition function a needs some certainty about the prediction, common choices are Gaussian processes [1, 23, 25, 33] or ensembles, such as random forests [12]. Typical acquisition functions are the expected improvement [13], the probability of improvement [13], the conditional entropy of the minimizer [28] or a multi-armed bandit based criterion [24]. The expected improvement is the most prominent choice for hyperparameter optimization and is also the acquisition function, which we choose. Formally, the improvement for a hyperparameter configuration $\boldsymbol{\lambda}$ is defined as

$$I(\boldsymbol{\lambda}) = \max\left\{f^{\min} - Y, 0\right\} \tag{4}$$

where $f^{\min}$ is currently the best function value and Y is a random variable modeling our knowledge about the value of the function f for the hyperparameter configuration $\boldsymbol{\lambda}$, which depends on $\mathcal{H}_t$. The hyperparameter configuration with highest expected improvement, i.e.

$$\mathrm{E}\left[I(\boldsymbol{\lambda})\right] = \mathrm{E}\left[\max\left\{f^{\min} - Y, 0\right\} \mid \mathcal{H}_t\right], \tag{5}$$

is chosen for the next evaluation. Assuming $Y \sim \mathcal{N}\left(\mu\left(\Psi_{t+1}(\boldsymbol{\lambda})\right), \sigma^2\left(\Psi_{t+1}(\boldsymbol{\lambda})\right)\right)$, the expected improvement can be formulated in closed-form as

$$\mathrm{E}\left[I(\boldsymbol{\lambda})\right] = \begin{cases} \sigma\left(\Psi_{t+1}(\boldsymbol{\lambda})\right)\left(Z \cdot \Phi(Z) + \phi(Z)\right) & \text{if } \sigma\left(\Psi_{t+1}(\boldsymbol{\lambda})\right) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where

$$Z = \frac{f^{\min} - \mu\left(\Psi_{t+1}(\boldsymbol{\lambda})\right)}{\sigma\left(\Psi_{t+1}(\boldsymbol{\lambda})\right)} \tag{7}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the standard normal density and distribution function, and $\mu\left(\Psi_{t+1}(\boldsymbol{\lambda})\right)$ and $\sigma\left(\Psi_{t+1}(\boldsymbol{\lambda})\right)$ are the expected value and the standard deviation of the prediction $\Psi_{t+1}(\boldsymbol{\lambda})$.

4 Two-Stage Surrogate Model

Our proposed two-stage surrogate model is explained in this section. The first stage of the surrogate model approximates the hyperparameter response functions of a new data set and each data set from the meta-data individually with Gaussian processes. The second stage combines the first-stage models by taking the similarity between the new data set and the data set from previous experiments into consideration. We construct a ranking of hyperparameter configurations as well as a prediction about the uncertainty of this ranking. The proposed two-stage architecture is visualized in Figure 1.

4.1 Notation

In the following, the prefix *meta* is used to distinguish between the different learning problems. The traditional problem is to learn some parameters $\boldsymbol{\theta}$ on a given

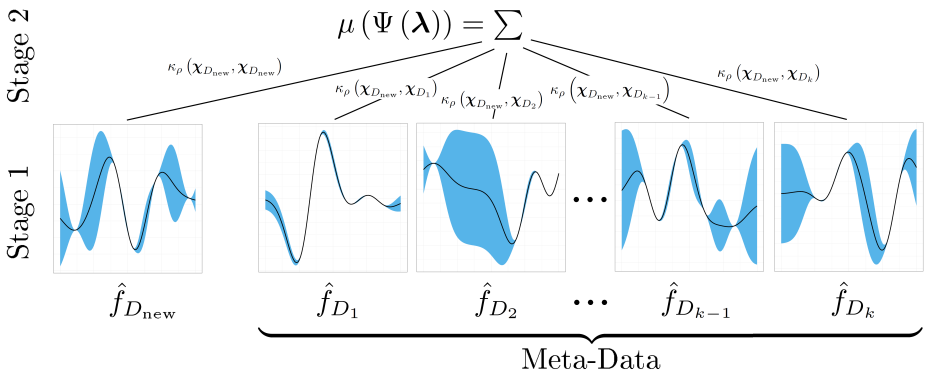

Fig. 1. At the first stage the hyperparameter response functions of the new data set D_{new} as well as data sets $\mathcal{D} = \{D_1, \ldots, D_k\}$ used for previous experiments are approximated using known evaluations. At the second stage the predictions of each individual model $\hat{f}_D$ are taken into account weighted by the similarity between D and D_{new} to determine the final predicted score.

data set containing instances with predictors. For the hyperparameter optimization problem you can create *meta-data sets* consisting of *meta-instances* with *meta-predictors*. A meta-data set contains meta-instances $(\boldsymbol{\lambda}_i, f_D(\boldsymbol{\lambda}_i))$ where $f_D(\boldsymbol{\lambda}_i)$ is the target and $\boldsymbol{\lambda}_i$ are the predictors. The hyperparameter response function $f_D : \Lambda \to \mathbb{R}$ is a function for a specific classifier and a specific data set D. For a given hyperparameter configuration, it returns the misclassification rate after training the classifier with the respective hyperparameter configuration on the training data set D. The task is to find a good hyperparameter configurations on a new data set D_{new} within T trials. To achieve this, a meta-data set, i.e. meta-instances for other data sets $D \in \mathcal{D}$, is given and this knowledge is transferred to the new problem.

4.2 First Stage - Hyperparameter Response Function Approximation

The first stage of our two-stage surrogate model approximates the hyperparameter response function for each data set. The meta-data set can be used to approximate the hyperparameter response function f_D for all $D \in \mathcal{D}$ by learning a machine learning model $\hat{f}_D$, using the meta-instances of each data set D. Similarly, we can learn an approximation $\hat{f}_{D_{\text{new}}}$ for the new data set, for which we have only few, but a growing number of meta-instances. Before learning $\hat{f}_D$ for all $D \in \mathcal{D}$, the labels of the meta-instances are scaled to $[0, 1]$ per data set. This is done such that each data set has equal influence on the second stage. The labels of the new data set D_{new} remain untouched.

For approximating the hyperparameter response function f_D, any machine learning model can be used, which is able to capture high non-linearity. We

decide to use Gaussian processes [19], which are a very prominent surrogate model for SMBO [1, 23, 25, 33].

4.3 Second Stage - Final Hyperparameter Configuration Ranking

The second stage combines all models of the first stage within one surrogate model Ψ to rank the different hyperparameter configurations and predict the uncertainty about the ranking. The predicted score of a hyperparameter configuration is determined using kernel regression [11]. We use the Nadaraya-Watson kernel-weighted average to predict the mean value of the surrogate model

$$\mu\left(\Psi\left(\boldsymbol{\lambda}\right)\right) = \frac{\sum_{D \in \mathcal{P} \cup \{D_{\text{new}}\}} \kappa_\rho\left(\boldsymbol{\chi}_{D_{\text{new}}}, \boldsymbol{\chi}_D\right) \hat{f}_D\left(\boldsymbol{\lambda}\right)}{\sum_{D \in \mathcal{P} \cup \{D_{\text{new}}\}} \kappa_\rho\left(\boldsymbol{\chi}_{D_{\text{new}}}, \boldsymbol{\chi}_D\right)} \tag{8}$$

with the Epanechnikov quadratic kernel

$$\kappa_\rho\left(\boldsymbol{\chi}_D, \boldsymbol{\chi}_{D'}\right) = \delta\left(\frac{\|\boldsymbol{\chi}_D - \boldsymbol{\chi}_{D'}\|_2}{\rho}\right) \tag{9}$$

with

$$\delta\left(t\right) = \begin{cases} \frac{3}{4}\left(1 - t^2\right) & \text{if } t \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $\rho > 0$ is the bandwidth and $\boldsymbol{\chi}_D$ is a vector describing the data set D. We discuss the description of data sets in-depth in the next section.

The predicted uncertainty for a hyperparameter configuration $\boldsymbol{\lambda}$ is defined as

$$\sigma\left(\Psi\left(\boldsymbol{\lambda}\right)\right) = \sigma\left(\hat{f}_{D_{\text{new}}}\left(\boldsymbol{\lambda}\right)\right) \tag{11}$$

Using Equations 8 and 11, the expected improvement for arbitrary hyperparameter configurations can be estimated. Thus, our Two-Stage Transfer surrogate model Ψ can be used within the SMBO framework described in Algorithm 1.

4.4 Data Set Description

In this section we introduce three different ways to describe data sets in vector form.

Description using Meta-Features The most popular way to describe data sets is by utilizing meta-features [1, 20, 22]. These are simple, statistical or information theoretic properties extracted from the data set. The similarity between two data sets, as defined in Equation 9, is then dependent on the Euclidean distance between the meta-features of the corresponding data sets. In this work we are using the meta-features listed in Table 1. For a more detailed explanation, we refer the reader to Michie et al. [17]. A well-known problem with meta-features is that it is a difficult problem to find and choose meta-features that are able to adequately describe a data set [15].

Table 1. The list of all meta-features used by us.

Meta-Features	
Number of Classes	Class Probability Max
Number of Instances	Class Probability Mean
Log Number of Instances	Class Probability Standard Deviation
Number of Features	Kurtosis Min
Log Number of Features	Kurtosis Max
Data Set Dimensionality	Kurtosis Mean
Log Data Set Dimensionality	Kurtosis Standard Deviation
Inverse Data Set Dimensionality	Skewness Min
Log Inverse Data Set Dimensionality	Skewness Max
Class Cross Entropy	Skewness Mean
Class Probability Min	Skewness Standard Deviation

Description using Pairwise Hyperparameter Performance Rankings Describing data sets based on pairwise hyperparameter performance rankings has been used in few approaches [16, 29]. The idea is to select all paired combinations of hyperparameter configurations $(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j)$ evaluated on the new data set D_{new} and estimate how often two data sets D and D' agree on the ranking. Usually, it is assumed that the hyperparameter configurations evaluated on D_{new} have also been evaluated on all data sets from the meta-data set $D \in \mathcal{D}$. In the context of general hyperparameter tuning, this is likely not the case. Therefore, we propose to use the first stage predictors to approximate the performances to overcome this problem.

Formally, given t many observations on D_{new} for the hyperparameter configurations $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t$, D_{new} can be described as $\boldsymbol{\chi}_{D_{\text{new}}} = ((\chi_{D_{\text{new}}})_i)_{i=1,\ldots,t^2} \in \mathbb{R}^{t^2}$,

$$\left(\boldsymbol{\chi}_{D_{\text{new}}}\right)_{j+(i-1)t} = \begin{cases} 1 & \text{if } f_{D_{\text{new}}}\left(\boldsymbol{\lambda}_i\right) > f_{D_{\text{new}}}\left(\boldsymbol{\lambda}_j\right) \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

Similarly, using the same t hyperparameter configurations, we can define for all $D \in \mathcal{D}$

$$\left(\boldsymbol{\chi}_{D}\right)_{j+(i-1)t} = \begin{cases} 1 & \text{if } \hat{f}_{D}\left(\boldsymbol{\lambda}_i\right) > \hat{f}_{D}\left(\boldsymbol{\lambda}_j\right) \\ 0 & \text{otherwise} \end{cases}. \tag{13}$$

Please note that we use $\hat{f}$ instead of f. As explained before, a hyperparameter configuration that is evaluated on D_{new} was likely not evaluated on all data sets $D \in \mathcal{D}$. For this reason we predict the performance using the first stage predictors. Using this description, the Euclidean distance between two data sets is the number of discordant pairs [14].

5 Experimental Evaluation

5.1 Tuning Strategies

We introduce all tuning strategies considered in the experiments. We consider strategies that do not use knowledge from previous experiments as well as those that use it.

Random Search. As the name suggests, this strategy chooses hyperparameter configurations at random. Bergstra and Bengio [3] have shown that this outperforms grid search in scenarios with hyperparameters with low effective dimensionality.

Independent Gaussian Process (I-GP). This tuning strategy uses a Gaussian process with squared-exponential kernel with automatic relevance determination (SE-ARD) as a surrogate model [23]. It only uses knowledge from the current data set and is not using any knowledge from previous experiments.

Spearmint. While I-GP is our own implementation of SMBO with a Gaussian process as a surrogate model we also compare to the implementation by Snoek et al. [23]. The main difference to I-GP is the use of the Matérn $5/2$ kernel. We added this as a baseline because it is considered to be a very strong baseline.

Independent Random Forest (I-RF). Besides Gaussian processes, random forests are another popular surrogate model [12], which we compare against in the experiments. We compared our own implementation against the original implementation of SMAC. Since our implementation provided stronger results, we will report these results. No knowledge from previous experiments is employed.

Surrogate Collaborative Tuning (SCoT). SCoT [1] uses meta-knowledge in a two step approach. In the first step, an SVMRank is learned over the whole meta-data. Its prediction for the meta-instances are used to replace the labels of the meta-instances. Bardenet et al. [1] argue that this overcomes the problem of having data sets with different scales of labels. On this transformed meta-data set, a Gaussian process with SE-ARD kernel is trained. In the original work it was proposed to use an RBF kernel for SVMRank. For reasons of computational complexity, we follow the lead of Yogatama and Mann [33] and use a linear kernel instead.

Gaussian Process with Multi-Kernel Learning (MKL-GP). Yogatama and Mann [33] propose to use a Gaussian process as a surrogate model for the SMBO framework. To tackle the problem of different scales on different data sets they are normalizing the data. Furthermore, they are using a kernel which is a linear combination of an SE-ARD kernel and a kernel modeling the distance between data sets.

Factorized Multilayer Perceptron (FMLP). FMLP [21] is the most recent surrogate model we are aware of. Published on last year's ECML PKDD, it is using a specific neural network to learn the similarity between the new data set and those from previous ones implicitly in a latent representation.

Two-Stage Transfer Surrogate (TST). This is the surrogate model proposed by us in this work. We consider two variations with two different data set representations. TST-M is using the meta-feature representation for the data sets, TST-R is using the pairwise ranking representation. We are using SE-ARD kernels for the Gaussian processes.

The kernel parameters are learned by maximizing the marginal likelihood on the meta-training set [19]. All hyperparameters of the tuning strategies are optimized in a leave-one-data-set-out cross-validation on the meta-training set.

The results reported estimated using a leave-one-data-set-out cross-validation and are the average of ten repetitions. For strategies with random initialization (Random, I-GP, Spearmint, I-RF), we report the average over thousand repetitions due to the higher variance. Hyperparameter configurations are limited to the precomputed grid which makes the experiment computational feasible for our infrastructure. We do not believe that limiting the black-box search to a grid has any impact on the results. In the end, this can be considered as additional constraints on the search space. In practice, our surrogate model allows finding arbitrary hyperparameter configurations like all other competitor methods. The evaluation was committed in the same way for transferring and non-transferring methods. Meta-hyperparameters for the surrogate models were individually tuned. For those strategies that use meta-features (SCoT, MKL-GP, TST-M), we use those meta-features that are described in Table 1.

5.2 Meta-Data Sets

We use two meta-data set introduced in [31] but increase the number of meta-features from three to the 22 listed in Table 1. The support vector machine (SVM) meta-data set was created using 50 classification data sets chosen at random from the UCI repository. Existing train/test splits were merged, shuffled and split into 80% train and 20% test.

The SVM [4] was trained for three different kernels (linear, polynomial and Gaussian) such that the hyperparameter dimension is six. Three dimensions are used for kernel indicator variables, one for the trade-off parameter C, one for the degree of the polynomial kernel d and one for the width γ of the Gaussian kernel. If a hyperparameter was not involved, its value was set to 0. The misclassification error was precomputed on the grid $C \in \left\{2^{-5}, \ldots, 2^{6}\right\}$, $d \in \{2, \ldots, 10\}$ and $\gamma \in \left\{10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5, 1, 2, 5, 10, 20, 50, 10^{2}, 10^{3}\right\}$ resulting into 288 meta-instances per data set. Creating this meta-data set took about 160 CPU hours.

Furthermore, we use a Weka meta-data set to evaluate on the combined algorithm and hyperparameter configuration problem as tackled in [27]. 59 classification data sets are preprocessed as done for the SVM meta-data set. Using 19 different Weka classifiers [10], we precomputed the misclassification error on a grid which resulted into 21,871 hyperparameter configurations per data set such that the overall meta-data contains 1,290,389 meta-instances. It took us more than 891 CPU hours to create this meta-data set.

To show that the tuning strategies can also deal with hyperparameter configurations they have never seen on other data sets, the tuning strategies only have access on meta-instances on a subset of the meta-instances. The evaluation on meta-test was done using all meta-instances.

To enable reproducibility, we provide a detailed description of the meta-data sets, the meta-data sets itself and our source code on GitHub [30].

5.3 Evaluation Metrics

We compare all tuning methods with respect to two common evaluation metrics: average rank and average distance to the global minimum. The average rank ranks the tuning strategies per data set according to the best found hyperparameter configuration. These ranks are then averaged over all data sets. The average distance to the global minimum after t trials is defined as

$$\text{ADTM}\left(\Lambda_t, \mathcal{D}\right) = \sum_{D \in \mathcal{D}} \min_{\boldsymbol{\lambda} \in \Lambda_t} \frac{f_D\left(\boldsymbol{\lambda}\right) - f_D^{\min}}{f_D^{\max} - f_D^{\min}} \tag{14}$$

where $f_D^{\max}$ and $f_D^{\min}$ are the worst and best value on the precomputed grid, respectively. Λ_t is the set of hyperparameter configurations, that have been evaluated in the first t trials. The performance per data set is scaled between 0 and 1 to get rid of the influence of different misclassification offsets and scales. Finally, the distances between the performance of the best performing hyperparameter configuration found to the best possible performance on the grid is averaged over all data sets.

5.4 Experiments

We compare the different hyperparameter optimization methods in two different scenarios: i) hyperparameter tuning and ii) combined algorithm selection and hyperparameter tuning. For the task of hyperparameter tuning, we optimize the hyperparameters of a support vector machine. The results are summarized in Figure 2. What we can see is that TST-R is outperforming the competitor methods with respect to both evaluation metrics by a large margin. TST-M has a similar good start as TST-R but its performance degenerates after few trials. Because the only difference between TST-R and TST-M is the way the data sets are described, one might argue that meta-features are less descriptive in describing a data set than the approach of pairwise rankings. We do not think that one can infer this from these results. The true reason for this behavior is

that the distances for TST-R are updated after each trials and the distance to the data sets from previous experiments is increasing over time. Thus, the influence of the meta-data set vanishes and TST-R is focusing only on the knowledge about the new data set at some point of time. Contrariwise, TST-M is using a constant distance between data set based on the meta-features. While the meta-knowledge is useful especially in the beginning, TST-M keeps relying on this such that the information of the new data set is not optimally taken into account. One simple way of fixing this problem is to decay the influence of the meta-knowledge which would introduce at least one meta-hyperparameter. Because TST-R is performing well without an additional meta-hyperparameter for the decay, we do not follow this idea here.

Spearmint provides stronger results than I-GP due to the choice of a different kernel. This might be an indication that we can further improve TST-R, if we use the Matérn 5/2 kernel instead of the SE-ARD.

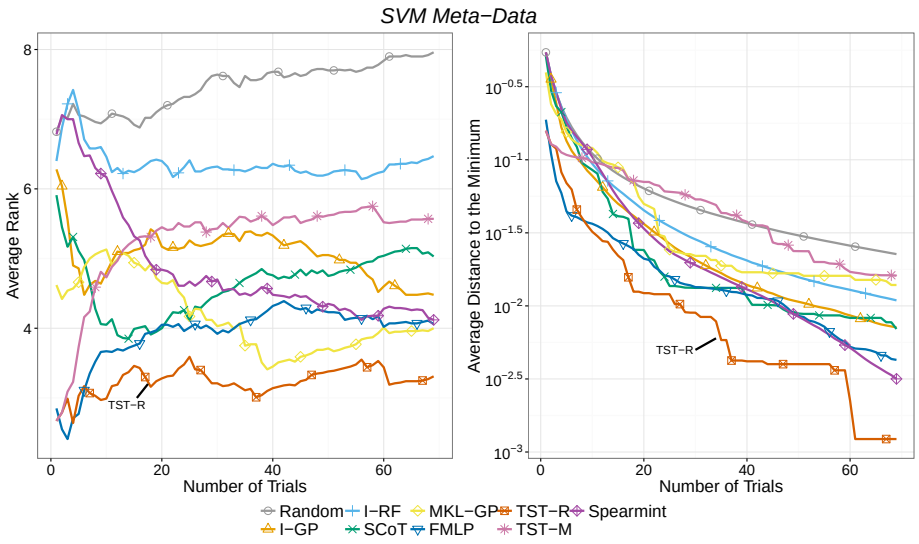

Fig. 2. Our proposed transfer surrogate model TST-R provides the best performance with respect to both evaluation measures for the task of hyperparameter tuning. For both metrics, the smaller the better.

We investigate the performance of the optimization methods also for the problem of combined algorithm selection and hyperparameter tuning on our Weka meta-data set. For this experiment, we remove some methods for different reasons. We remove some weaker methods (Random and TST-M) to improve the readability. Furthermore, we do not compare to methods, which are using one Gaussian process, that is trained on the complete meta-data (SCoT and MKL-GP). The reason for this is that Gaussian processes do not scale to these

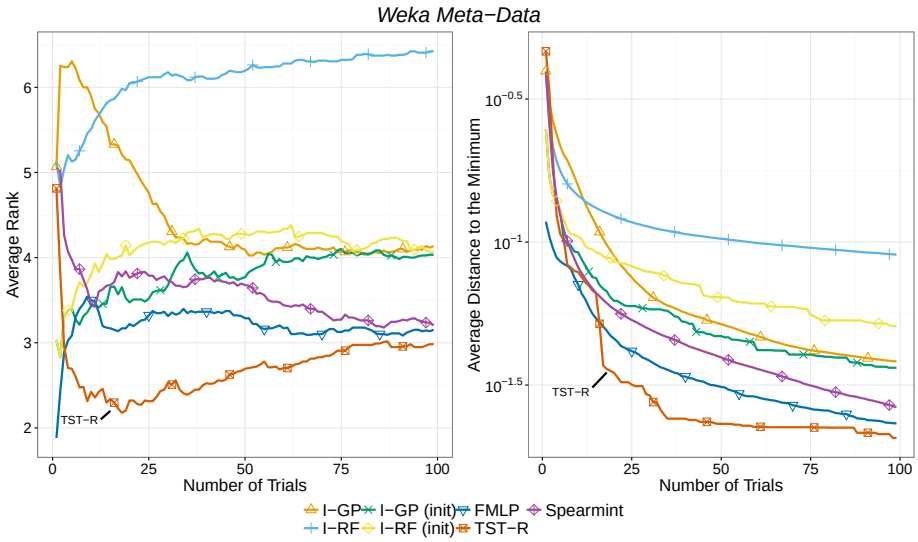

Fig. 3. Our approach TST-R also outperforms the competitor methods for the task of combined algorithm selection and hyperparameter tuning. Surrogate models that use Gaussian processes that train over the whole meta-data are not feasible for this data set [31]. Therefore, we consider I-GP and I-RF with meta-learning initialization.

large meta-data sets (time and memory-wise) [31]. Our approach is learning one Gaussian process for each data set such that each model only needs to be learned on a fraction of the data and thus remains feasible. Nevertheless, we compare to FMLP, the strongest competitor from the previous experiment as well as I-GP and I-RF. Furthermore, we also compare to I-GP and I-RF with five initialization steps using a strong meta-initialization technique [32]. The results summarized in Figure 3 are very similar to our previous experiment. TST-R again is best for both evaluation metrics but FMLP shows to be a strong competitor.

6 Conclusion

In this work, we propose a two-stage transfer surrogate for using meta-knowledge to accelerate the search with the SMBO framework. We propose to approximate the hyperparameter response surface of each data set with an individual model. These individual models are finally combined at the second stage to estimate the score of a hyperparameter configuration. In extensive experiments on two meta-data sets, we compare our method to numerous competitor methods published recently on established machine learning conferences. We show empirically that our two-stage transfer surrogate model is able to outperform all considered competitor methods for the task of hyperparameter tuning as well as the task of combined algorithm selection and hyperparameter tuning.

For future work we are planning to have a deeper look into different ways of describing data sets. Furthermore, we want to investigate whether it is possible to add a decay meta-hyperparameter that enables our approach to also work with typical data set descriptions such as meta-features. Most importantly, we want to investigate the impact of different kernels for TST on the performance. Currently, the Matérn 5/2 seems to be a promising candidate.

Acknowledgments. The authors gratefully acknowledge the co-funding of their work by the German Research Foundation (DFG) under grant SCHM 2583/6-1.

References

1. Bardenet, R., Brendel, M., Kégl, B., Sebag, M.: Collaborative hyperparameter tuning. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. pp. 199–207 (2013)
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain. pp. 2546–2554 (2011)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305 (Feb 2012)
4. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
5. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. pp. 215–223 (2011)
6. Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 3460–3468 (2015)
7. Feurer, M., Springenberg, J.T., Hutter, F.: Using meta-learning to initialize bayesian optimization of hyperparameters. In: ECAI workshop on Metalearning and Algorithm Selection (MetaSel). pp. 3–10 (2014)
8. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 1128–1135 (2015)
9. Gomes, T.A.F., Prudêncio, R.B.C., Soares, C., Rossi, A.L.D., Carvalho, A.C.P.L.F.: Combining meta-learning and search techniques to select parameters for support vector machines. Neurocomputing 75(1), 3–13 (2012)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explor. Newsl. 11(1), 10–18 (Nov 2009)
11. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction. Springer, 2 edn. (2009)

12. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proceedings of the 5th International Conference on Learning and Intelligent Optimization. pp. 507–523. LION'05, Springer-Verlag, Berlin, Heidelberg (2011)
13. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. of Global Optimization 13(4), 455–492 (Dec 1998)
14. Kendall, M.G.: A New Measure of Rank Correlation. Biometrika 30(1/2), 81–93 (Jun 1938)
15. Leite, R., Brazdil, P.: Predicting relative performance of classifiers from samples. In: Proceedings of the 22Nd International Conference on Machine Learning. pp. 497–503. ICML '05, ACM, New York, NY, USA (2005)
16. Leite, R., Brazdil, P., Vanschoren, J.: Selecting classification algorithms with active testing. In: Machine Learning and Data Mining in Pattern Recognition - 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings. pp. 117–131 (2012)
17. Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J. (eds.): Machine Learning, Neural and Statistical Classification. Ellis Horwood, Upper Saddle River, NJ, USA (1994)
18. Pinto, N., Doukhan, D., DiCarlo, J.J., Cox, D.D.: A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLoS Computational Biology 5(11), e1000579 (2009), PMID: 19956750
19. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
20. Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. Machine Learning 87(3), 357–380 (2012)
21. Schilling, N., Wistuba, M., Drumond, L., Schmidt-Thieme, L.: Hyperparameter optimization with factorized multilayer perceptrons. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015. Proceedings, Part II (2015)
22. Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. 41(1), 6:1–6:25 (Jan 2009)
23. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 2960–2968 (2012)
24. Srinivas, N., Krause, A., Kakade, S., Seeger, M.W.: Gaussian process optimization in the bandit setting: No regret and experimental design. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. pp. 1015–1022 (2010)
25. Swersky, K., Snoek, J., Adams, R.P.: Multi-task bayesian optimization. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 2004–2012 (2013)
26. Swersky, K., Snoek, J., Adams, R.P.: Freeze-thaw bayesian optimization (2014)
27. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 847–855. KDD '13, ACM, New York, NY, USA (2013)

28. Villemonteix, J., Vazquez, E., Walter, E.: An informational approach to the global optimization of expensive-to-evaluate functions. Journal of Global Optimization 44(4), 509–534 (2009)
29. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Sequential model-free hyperparameter tuning. In: Data Mining (ICDM), 2015 IEEE International Conference on. pp. 1033–1038 (Nov 2015)
30. Wistuba, M.: Supplementary website: "https://github.com/wistuba/TST" (Mar 2016)
31. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Hyperparameter search space pruning - a new component for sequential model-based hyperparameter optimization. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015. Proceedings, Part II (2015)
32. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Learning hyperparameter optimization initializations. In: International Conference on Data Science and Advanced Analytics, DSAA 2015, Paris, France, October 19 - 21, 2015 (2015)
33. Yogatama, D., Mann, G.: Efficient transfer learning method for automatic hyperparameter tuning. In: International Conference on Artificial Intelligence and Statistics (AISTATS 2014) (2014)